Reproduce, Replicate, Reevaluate. The Long but Safe Way to Extend Machine Learning Methods

Luisa Sophie Werner, Nabil Layaïda, Pierre Genevès, Jérôme Euzenat, Damien Graux

▶ **To cite this version:**

Luisa Sophie Werner, Nabil Layaïda, Pierre Genevès, Jérôme Euzenat, Damien Graux. Reproduce, Replicate, Reevaluate. The Long but Safe Way to Extend Machine Learning Methods. AAAI 2024 - 38th Annual AAAI Conference on Artificial Intelligence, Feb 2024, Vancouver, Canada. pp.1-9. ⟨hal-04035305v4⟩

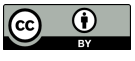
HAL Id: hal-04035305

https://inria.hal.science/hal-04035305v4

Submitted on 13 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

Reproduce, Replicate, Reevaluate.
The Long but Safe Way to Extend Machine Learning Methods

Luisa Werner[1], Nabil Layaïda[1], Pierre Genevès[1], Jérôme Euzenat[1], Damien Graux[2]

[1]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG F-38000 Grenoble, France.
[2]ADAPT SFI Research Centre, Trinity College Dublin, Ireland.
{Luisa.Werner, Nabil.Layaida, Pierre.Geneves, Jerome.Euzenat}@inria.fr, grauxd@tcd.ie

Abstract

Reproducibility is a desirable property of scientific research. On the one hand, it increases confidence in results. On the other hand, reproducible results can be extended on a solid basis. In rapidly developing fields such as machine learning, the latter is particularly important to ensure the reliability of research. In this paper, we present a systematic approach to **reproducing** (using the available implementation), **replicating** (using an alternative implementation) and **reevaluating** (using different datasets) state-of-the-art experiments. This approach enables the early detection and correction of deficiencies and thus the development of more robust and transparent machine learning methods. We detail the independent reproduction, replication, and reevaluation of the initially published experiments with a method that we want to extend. For each step, we identify issues and draw lessons learned. We further discuss solutions that have proven effective in overcoming the encountered problems. This work can serve as a guide for further reproducibility studies and generally improve reproducibility in machine learning.

Introduction

Experiment reproducibility is a cornerstone of scientific research. It is so for the experimenters, because it allows them to be more confident in the results they claim. It is so for the research community, because it allows other researchers to stand on a solid ground in developing their own work. This leads to more reliable and superior research results, which is beneficial for the society as a whole.

In the machine learning community, strong incentives emerge to provide the necessary information to reproduce results (Pineau et al. 2021). More generally, this is a basic open science requirement to increase accountability and reproducibility. Reproducibility guidelines and checklists focus on encouraging researchers to provide more detailed documentation of their published work (IJCAI 2022; ICML 2023; AAAI 2023). However, it is not obvious that such guidelines are sufficient to independently reproduce experiments. Indeed, random operations may be introduced in several places (Chen et al. 2022), hyperparameters may change the behavior of models (Henderson et al. 2018; Lucic et al. 2018), components may exchange information in imprecise ways, the data processing pipeline may be overlooked, etc. (Lones 2023; Forde and Paganini 2019). All these factors can compromise the reproducibility of experiments.

	Same software	Different software
Same datasets	**Reproduce**	**Replicate**
Different datasets	**Reevaluate**	

Table 1: Overview of the three steps.

Various terminologies have been considered for reproducibility in computer science (Drummond 2009; Goodman, Fanelli, and Ioannidis 2016; Rougier et al. 2017; Plesser 2018; Gundersen, Gil, and Aha 2018). We use and enrich the terminology adopted by the ACM (ACM 2016), as summarized in Table 1:

- **'repeat'** means reexecuting an experiment with the same code, the same parameters, the same data by the same experimenter;
- **'reproduce'** means performing an experiment with the same software, the same parameters and the same data but by a different experimenter;
- **'replicate'** means performing an experiment independently on the same data but using different software and by a different experimenter;
- to these, we add **'reevaluate'** which means performing an experiment independently on different data.

When necessary, 'reproducibility' is also used as a term covering all such activities.

This work is motivated by our objective to extend the Knowledge Enhanced Neural Networks (KENN) (Daniele and Serafini 2023) to knowledge graphs. To that extent, we need to reimplement the system, available in Tensorflow (Abadi et al. 2015), in the graph-specific library PyTorch Geometric (Fey and Lenssen 2019). This is a common use case as, beyond the motivation to extend, there may be various other reasons to reimplement a system (Tian et al. 2022; Alahmari et al. 2020). For that purpose, we provide a progressive approach for extending a machine learning method by first going through reproducing, replicating and reevaluating its results. This approach aims at ensuring that the reimplementation is reliable. It is tempting to skip these steps and jump directly to method extension. On the contrary, we argue that this approach allows re-

searchers to better understand potential problems at the specific stage where they occur. Further, we identify the obstacles encountered when conducting these steps and how they may be overcome. The particular case on which we detail and demonstrate our approach allows us to derive general lessons learned. Although we do not claim that this work can solve all reproducibility problems, it contributes to improve reproducibility in the field of machine learning.

Related Work

Given the high degree of non-determinism of machine learning techniques in general, the problem of reproducing results has gained importance with the increase of interest in the field. Recent surveys (Gundersen and Kjensmo 2018) show that even papers published at prestigious conferences are not sufficiently documented and report on a reproducibility crisis (Gundersen 2020). An empirical study (Raff 2019) reveals difficulties in independently replicating published papers. Due to increased awareness, several conferences, including AAAI, promote reproducibility by encouraging authors to provide source code, experimental descriptions and datasets of their papers through reproducibility guidelines and checklists (Pineau et al. 2021; IJCAI 2022; AAAI 2023).

Further, there has been a surge in documented reproduction and replication attempts, particularly encouraged through reproducibility challenges (Pineau et al. 2021; Sinha et al. 2022; Manninen et al. 2018). Such challenges ask interested individuals to replicate results from recent papers. Difficulties such as the lack of detailed documentation, e.g. hyperparameters (Ankit et al. 2022) or even failure to reproduce (Blanco et al. 2020) are frequently reported. Their motivation differs from ours, as our goal is not to achieve reproducibility for its own sake. We reproduce experiments to ensure that the system we reimplement retains the function of the initial implementation.

Knowledge Enhanced Neural Networks

To overcome the limitations of deep learning such as data-hungriness or black-box properties, the research field of neuro-symbolic integration aims at combining neural methods with symbolic AI (Garnelo and Shanahan 2019). KENN is a neuro-symbolic method that integrates a base neural network with knowledge enhancement layers in an end-to-end differentiable way. The knowledge enhancement layers encode prior knowledge in form of first-order logic clauses and revise the predictions of the neural network with respect to the satisfaction of the prior knowledge. Fuzzy logic is used to represent first-order logic clauses in the real-valued domain. A knowledge enhancement layer contains *clause weights* as trainable parameters which are jointly optimized with the parameters of the base neural network and quantify the importance of a clause. By introducing binary predicates in its logic language, KENN (Daniele and Serafini 2023) can model graph data and has been applied to a node classification task on the Citeseer dataset (Lu and Getoor 2003). The Citeseer dataset consists of scientific papers belonging to one of the six computer science research classes and citations among them. The dataset can be modelled as

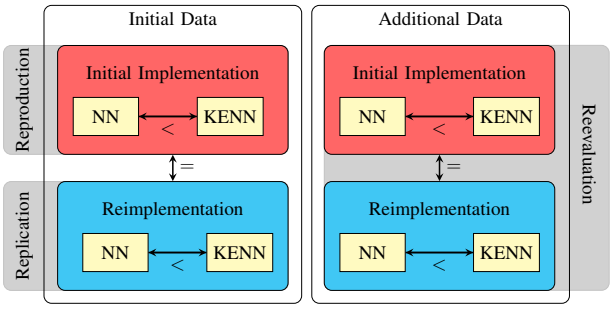


Figure 1: Overview of the approach.

a graph where papers are represented by nodes and citations by edges. The prior knowledge provided to the knowledge enhancement layers is derived from the assumption that two papers that cite each other have the same class. According to the pattern $\forall x \forall y : \neg Cls(x) \vee \neg Cite(x; y) \vee Cls(y)$, one logical clause is defined for each class (Cls) to obtain a prior knowledge base. While the predictions of the base neural network are used as interpretation of the unary predicates representing the classes in the real-valued domain, the citations between two papers are known a priori. For this reason, the predictions of the binary predicate $Cite(x, y)$ are set to true (1.) if a citation exists.

Because we aim at extending KENN with more graph learning capabilities, we need to be sure that our work stands on firm ground. For that purpose, we reproduce the experiments performed by the authors.

In (Daniele and Serafini 2023), the authors report the performance of KENN for experiments on multiple training set sizes (10%, 25%, 50%, 75%, 90%). For each size, 100 independent runs are conducted. KENN is compared to the standalone base neural network and to Relational Neural Machines (RNM) (Marra et al. 2020) and Semantic-Based Regularization (SBR) (Diligenti, Gori, and Saccà 2017). The experiments support the following hypotheses:

H1 KENN consistently outperforms the base neural network for all training set sizes.

H2 The performance gain due to the knowledge enhancement is larger when training data is limited.

H3 KENN leads to similar or superior results compared to RNM and SBR.

Methodology

We extend KENN in the framework PyTorch (Paszke et al. 2019) in conjunction with the library PyTorch Geometric. To this end, a comprehensive reproducibility study is the best way to ensure that KENN is extended on a reliable basis. With this objective in mind, as illustrated in Fig. 1, we conduct the following steps:

1. We **reproduce** the results obtained with the initial implementation;
2. We reimplement the initial experiments and **replicate** them;
3. We **reevaluate** the experiments on additional datasets.

For each step, we detail the required information, the obstacles we had to overcome and how we managed to overcome them. Further, we evaluate whether the KENN results are confirmed and report the lessons learned.

We refer to the following publicly available material. The reproduction and replication is based on the results reported in the paper (Daniele and Serafini 2023) (that we call the *initial paper*) and on the reported experiments[1] (that we call the *initial experiments*). The experiments make use of the Python package KENN2[2,3]. In addition, we also extract information from other related papers (Daniele and Serafini 2019, 2020). We preserve, as far as possible, the input (dataset) and output format (results) of the initial experiment. In this work, we focus on the transductive part of the initial experiments with KENN. In the following, we refer to our implementation as *reimplementation*.

All experiments in this work are conducted on a machine running an Ubuntu 20.4 equipped with an Intel(R) Xeon(R) Silver 4114 CPU 2.20GHz processor, 192G of RAM and one GPU Nvidia Quadro P5000.

Evaluation Criteria

In order to judge whether we have successfully reproduced, the question of evaluating reproducibility arises. For this purpose we define two reproducibility targets.

As common in natural sciences, **qualitative reproducibility** refers to checking whether experimental results support the hypotheses of the initial paper and come to the same conclusions. Coming rather from an engineering perspective, **quantitative reproducibility** aims at obtaining the same or close results as the initial experiment. In Fig. 1, they are respectively noted as $>$ and $=$. In the context of the experiments with KENN, qualitative reproducibility refers to checking if the confirmed hypotheses in the initial paper are still supported by the reimplementation.

Regarding qualitative reproducibility, we focus on the hypotheses $H1$ and $H2$. The verification of $H3$ would involve reproducing the results of the baselines SBR and RNM which is beyond the scope of this work. To test $H1$ and $H2$, we adopt the procedures used in the initial paper. For $H1$, a one-sided independent Student t-test with significance threshold 0.01 is employed to assess the superiority of the mean accuracy of KENN over the base neural network. For $H2$, which establishes a relation between the training set size and the delta of base neural network and KENN, no precise evaluation procedure is mentioned. However, the absolute difference between the mean test accuracies for the experiments is observed.

Regarding the quantitative reproducibility, absolute equality of the results is hard to be expected, since even reexecuting the initial implementation does not lead to absolutely equal numbers. This can be due to differences in software or hardware or the absence of fixed seeds (Chen et al. 2022; Pham et al. 2021; PyTorch 2022). Therefore, we evaluate the steps of replication and reevaluation by examining the distributions of the reported test results and in particular their similarity. In order to test their equality, we compute the two-sided Kolmogorov-Smirnov goodness-of-fit test (KS-test) (Massey 1951), which checks whether two samples are drawn from the same distribution. If the p-value is below the significance threshold, there is evidence that the results of the experiments are not drawn from the same distribution. It has to be mentioned that the KS-test can only provide significant results in measuring inequality.

	Reported results			**Initial Implementation**		
train	NN	KENN	Delta	NN	KENN	Delta
10%	0.544	0.652	0.108	0.540 (0.067)	0.651 (0.017)	0.110 (0.067)
25%	0.629	0.702	0.073	0.630 (0.018)	0.702 (0.012)	0.072 (0.012)
50%	0.680	0.744	0.065	0.681 (0.187)	0.745 (0.012)	0.064 (0.021)
75%	0.733	0.788	0.055	0.733 (0.025)	0.791 (0.016)	0.058 (0.026)
90%	0.759	0.808	0.049	0.758 (0.027)	0.807 (0.022)	0.049 (0.028)

Table 2: Reproduction results.

Reproduction

As a first step, we reproduce the initial experiments with KENN. Therefore, we need (*a*) the executable, (*b*) the dataset, (*c*) the instructions and environment information to run the executable, (*d*) the procedure to collect and interpret the results. Given these elements, it is possible to process the instructions to reproduce the experiment.

Pitfalls and Workarounds

The paper (Daniele and Serafini 2020) provides the KENN2[2] package. It contains the implementation of the knowledge enhancement layers in Tensorflow but not the experiments. Through an exchange with the authors, we gained access to the source code and data of the experiments which were made publicly available[1]. They also pointed to another paper (Daniele and Serafini 2023) which further documents the experiments. The hyperparameters in the public implementation correspond to the results reported in (Daniele and Serafini 2023) but not to the results found in (Daniele and Serafini 2020).

The repository provides (*a*) a reference to the KENN2 package, (*b*) a link to the dataset, (*c*) a README file with instructions to run the code, and (*d*) a Jupyter notebook to analyze the results. The initial experiments use a dataset that is included in the repository instead of referring to an external, widely available version of the dataset. A link to the data source was provided, but it was inaccessible. Neither the origin of the data nor any filtering or preprocessing steps were documented. The instructions to run the experiments are reasonably complete, including a description of the required Python modules as a requirements file and the full command line to be run. However, the requirements only contain a list of packages without their version number. This can be a critical concern as Python packages evolve frequently, sometimes compromising compatibility. The Python version is not defined either which can be crucial for the successful execution of the experiments. We inferred it from the descrip-

[1]https://github.com/rmazzier/KENN-Citeseer-Experiments

[2]https://github.com/DanieleAlessandro/KENN2

[3]https://github.com/HEmile/KENN-PyTorch

tion of the KENN2 package. In both papers (Daniele and Serafini 2020, 2023) experimental results are reported in numeric tables. Despite seeded random operations, we observe subtle variations in the results. We note a non-deterministic set operation which may introduce variation. As a result, exact quantitative reproducibility cannot be guaranteed.

Results

After achieving a fully functional environment setup, we successfully run the experiments. The obtained results are reported in Table 2. In comparison to the results initially provided (Daniele and Serafini 2023), we add information such as the standard deviation of the test accuracies over all runs and the p-values for a t-test.

Qualitative Reproducibility For $H1$, KENN significantly outperforms the base neural network for all training set sizes ($p \ll 0.01$) which is consistent with the authors' claims. Regarding $H2$, we observe that the reported difference (column delta) in mean accuracy between KENN and the base neural network is larger for smaller training set sizes. Note that, since the initial experiments use paired samples, the means of the deltas can be reported, though we can only report the difference of the means.

Quantitative Reproducibility The reported and reproduced results cannot be compared statistically, since the full sample of the reported results is not available.

Lessons Learned

L1: Provide a Precise Identification of the Software Environment, Source Code and Dataset A first lesson learned is that the complete description of the software environment including version numbers of all modules should be provided. Furthermore, the datasets used should be described precisely, including information on their origin and on any applied preprocessing. Both are critical information to reproduction.

L2: Automate the Steps for Reproduction A second lesson learned is that the reproduction should be made as automatic as possible by the initial authors. Such an automation would even be useful for the initial authors to be able to repeat their experiments routinely.

L3: Provide Contact Information and Be Reachable The reachability and assistance of the authors was essential to clarify open questions and to get access to key components for reproducibility such as the experiment source code.

L4: Distinguish Different Experiments We missed a unique reference to the experiments. In fact, the authors published a first version of a paper (Daniele and Serafini 2019) and then several others (Daniele and Serafini 2023, 2020, 2023) which contain improvements and additional experimental settings. Still, all papers point to the same repository. This complicates the reproduction of the experiments of each paper. It would be a good practice to specify per paper the version of the repository or to use another repository.

	Reimplementation			
train	NN	KENN	Delta of means	p-values (KS Test)
10%	0.550 (0.042)	0.629 (0.076)	0.079	0.001
25%	0.632 (0.016)	0.676 (0.088)	0.044	0.024
50%	0.681 (0.014)	0.741 (0.028)	0.060	0.583
75%	0.729 (0.022)	0.785 (0.039)	0.056	0.054
90%	0.748 (0.026)	0.806 (0.020)	0.058	0.702

Table 3: Replication results.

Replication

In this section, we refer to the reimplementation in PyTorch and the replication of the experiments. Again, we compare the obtained results to the results of the published and reproduced experiments. To reimplement the system, we first identify the main components of the method by examining, on the one hand, the concepts of KENN introduced in the initial paper. On the other hand, we examine the code of the initial implementation to recover necessary information that is underspecified or not explicitly mentioned in the paper. As main components we identify (*a*) the data preprocessing, (*b*) the model definition, (*c*) the training loop, and (*d*) the hyperparameter definition. We reimplement these components as follows.

Data Preprocessing. We use the same Citeseer dataset and prior knowledge as provided for the initial experiments. We preserve the data splitting procedure, as well as the rebalancing of the prediction classes.

Model Definition. The KENN model consists of two stacked components: The base neural network and the knowledge enhancement layers. In the initial implementation, the base neural network is implemented with Keras (Chollet et al. 2015) and the knowledge enhancement layers are imported from the KENN2[2] package written in Tensorflow. We reimplement both the base neural network and use the knowledge enhancement layers from the KENN2 package[3].

Training Loop. In the training loop, we replace the optimizer and the loss function implemented in Tensorflow by their PyTorch equivalent.

Hyperparameter Definition. We identify the hyperparameters in the experiments and their values. To keep track of hyperparameters in a clean manner, we connect the reimplementation to an experiment tracking tool (Biewald 2020).

Pitfalls and Workarounds

At first sight, the reimplementation in PyTorch seems straightforward. However, we struggle in identifying the hyperparameters used in the initial implementation. The relevant information has to be gathered from various sources. The set of revealed hyperparameters, their assignments and how we recovered them is summarized in Table 4.

A first subset of hyperparameter values is explicitly named in the initial paper including the architecture of the base neural network, the initialization of the clause weights

Parameter	Value	In Paper	In Code	Defaults
NN - Number of Hidden Layers	3	✓	✓	
NN - Number of Hidden Neurons	50	✓	✓	
NN - Hidden Layer Activation	ReLu	✓	✓	
NN - Output Layer Activation	Linear	✓	✓	
KENN - Clause Weight Initialization	Constant, 0.5	✓	✓	
KENN - Binary Preactivations	500	✓	✓	
KENN - Number of KE Layers	3		✓	
KENN - Range Constraint	$[0.0, 500.0]$		✓	
Epochs	300		✓	
Batch Size	Full-batch		✓	
Loss function	Categorical Cross-entropy		✓	
Early Stopping - Patience	10		✓	
Early Stopping - Min Delta	0.001		✓	
Learning Rate	0.001			✓
NN - Weight Initialization	Random, Glorot uniform			✓
NN - Bias Initialization	Constant, Zeroes			✓
Optimizer	Adam (Kingma and Ba 2015), $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-7}$			✓
Dropout Rate	0.0, no dropout			

Table 4: The set of hyperparameters used in the initial experiments.

and the binary preactivation value. An additional subset of parameters is defined in a user modifiable script in the repository of the initial implementation. These script contains information on early stopping (and related parameters), the number of epochs, as well as the size of the validation set. Additional parameter values are required that are not mentioned in the paper, nor in the code documentation. By reviewing the source code, we recover the number of knowledge enhancement layers and the batch size, for example. An additional subset of hyperparameters is implicitly introduced and defined by framework-specific functions and their default assignments. These default values are found in the Tensorflow software documentation. From there, we recover the weight initialization of the dense layers in the base neural network and the optimizer-specific parameters.

Determining the values of some additional parameters turns out to be even more challenging. To get useful insights, we examine the isolated behavior of each component to ensure that they produce the same output, given some input. While randomness is essential to training neural networks, it complicates the analysis of the algorithm behavior. For the inspection of some components, we temporarily simplify operations and replace random numbers by fixed values. This allows to identify that linear layers are by default differently initialized in Tensorflow and in PyTorch. In Tensorflow, they are initialized randomly following a Glorot uniform distribution and the bias is initialized with zeroes. However, in PyTorch the weights and biases are randomly initialized following the uniform distribution.

Results

After recovering the full set of required parameter values and completing the reimplementation of KENN, we now assess the replicability of experiments. The results of the replication are summarized in Table 3.

Qualitative Reproducibility We first check whether $H1$ and $H2$ hold in the replicated experiment. We observe that the p-values of the t-test comparing the test accuracies for all training set sizes fall below the significance threshold in favor of $H1$ which is thus supported by the replication. For $H2$, considering the deltas of the mean test accuracies across the training set sizes, no clear relationship between training set size and its effect on the knowledge enhancement is apparent. This may be due to a reproducibility failure or to the lack of an explicit procedure to test $H2$ in the initial work. In conclusion, we are able to confirm $H1$, but we are not able to replicate the results concerning $H2$.

Quantitative Reproducibility The average accuracies of both implementations are relatively close (the highest observed difference is 0.02). However, their distributions are not identical. The similarity of the test accuracy distributions is computed against the test accuracy of the reproduced results, since we do not have the initial results from the authors of KENN. The p-values of the KS-test are listed in the right column. For the 10% training size, a significant ($p < 0.01$) difference between the distributions of the reproduced and replicated results is detected. For the remaining training set sizes, no difference appears significant. Hence, we consider our results as reasonably similar to those of the initial implementation, at least for large sample sizes. The accuracy obtained with 10% training set size appears less variable than the one on larger training sets.

Lessons Learned

L5: Document Hyperparameters Exhaustively The identification of all hyperparameters together with their assignments is critical for replication as they can considerably affect the results and thus the conclusions drawn from them. While some hyperparameters such as learning rate and batch size are standard in deep learning methods, custom models often define their own hyperparameters, such as KENN's

	Initial Implementation			Reimplementation			
train	NN	KENN	Delta	NN	KENN	Delta of means	p-values (KS-Test)
10%	0.530 (0.029)	0.750 (0.017)	0.220 (0.030)	0.576 (0.016)	0.766 (0.010)	0.190	$8.9\ 10^{-4}$
25%	0.606 (0.018)	0.800 (0.012)	0.193 (0.016)	0.647 (0.009)	0.819 (0.012)	0.173	$8.4\ 10^{-10}$
50%	0.652 (0.013)	0.833 (0.009)	0.187 (0.017)	0.678 (0.009)	0.831 (0.013)	0.152	$5.9\ 10^{-1}$
75%	0.691 (0.016)	0.850 (0.014)	0.159 (0.018)	0.686 (0.013)	0.833 (0.015)	0.147	$2.9\ 10^{-4}$
90%	0.715 (0.027)	0.871 (0.016)	0.156 (0.028)	0.743 (0.012)	0.913 (0.017)	0.170	$9.2\ 10^{-11}$

Table 5: Reevaluation results on the Cora dataset.

	Initial Implementation			Reimplementation			
train	NN	KENN	Delta	NN	KENN	Delta of means	p-values (KS-Test)
10%	0.333 (0.096)	0.405 (0.002)	0.071 (0.096)	0.326 (0.098)	0.404 (0.003)	0.077	$3.9\ 10^{-1}$
25%	0.341 (0.108)	0.416 (0.002)	0.075 (0.108)	0.380 (0.080)	0.414 (0.004)	0.034	$7.1\ 10^{-2}$
50%	0.409 (0.095)	0.443 (0.004)	0.033 (0.096)	0.364 (0.133)	0.441 (0.005)	0.077	$1.3\ 10^{-1}$
75%	0.447 (0.143)	0.498 (0.004)	0.051 (0.143)	0.414 (0.176)	0.495 (0.007)	0.081	$1.3\ 10^{-1}$
90%	0.505 (0.011)	0.510 (0.008)	0.004 (0.011)	0.504 (0.015)	0.504 (0.015)	-0.0001	$5.9\ 10^{-1}$

Table 6: Reevaluation results on the Pubmed dataset.

clause weights. Since they affect different components, hyperparameters are often declared at different stages in the experiment, which complicates their identification. In particular, some hyperparameters are implicitly defined as default parameters in the used library and are easily overlooked. A complete configuration file with the exhaustive list of hyperparameters, their description, and their value is key to replicability.

L6: Provide Clear Procedures to Check Claims To evaluate qualitative reproducibility, we rely on the authors' procedures for verifying their hypotheses. With respect to $H2$, the lack of a precise procedure complicates the evaluation of the replicability. Clear procedures for the verification of claims are essential to ensure their replicability.

L7: Define Standards to Evaluate Reproducibility In order to evaluate quantitative reproducibility, we use the KS-test that can only give statistical evidence that two distributions are not equal. Other metrics are proposed in (Piantadosi, Marrone, and Sansone 2019). To the best of our knowledge, no community standard on the evaluation of quantitative reproducibility exists. A community-agreed standard is needed to determine when two distributions of results can be considered equivalent.

Reevaluation

Having reproduced and replicated the experiments with KENN, we now reevaluate the method on two more datasets Cora and Pubmed (Yang, Cohen, and Salakhutdinov 2016). The goal of this reevaluation is (1) to check if the implementations behave robustly in the same way on different datasets and (2) to evaluate whether the hypotheses for KENN are valid on other datasets. These datasets are also citation graphs. While Cora and Citeseer have a similar size, Pubmed is significantly larger. Similarly to the experiments on Citeseer, the prior knowledge encodes a relation of paper category and citations. To avoid variations in the results due to hyperparameters, we use the set identified in Table 4 for all datasets. In general, hyperparameters should be determined separately for each dataset in order to obtain models with the best possible prediction quality. However, we are mainly concerned with obtaining the same results instead of the best possible results.

Results

The results for Cora and Pubmed are shown in Table 5 and 6. When applying KENN to Pubmed, we encounter a performance issue with the data preprocessing in the initial implementation. We modify the preprocessing to improve its scalability while maintaining its functionality.

Qualitative Reproducibility $H1$ is supported in both implementations ($p \ll 0.01$) for Cora. Concerning $H2$, the deltas in the initial implementation indicate a relationship between decreasing training set size and knowledge enhancement. However, this difference is rather uncertain for the reimplementation. For $H1$ on PubMed, the p-values of the t-tests are not below the significance threshold across all training set sizes, for the initial implementation only for training set sizes 10% and 25% and for the reimplementation for training set sizes 10%, 50%, 75% and 90%. Hence, no significant performance is gained with KENN. Furthermore, no monotonically increasing benefit of knowledge enhancement for smaller training set sizes can be observed with both implementations, therefore $H2$ is not supported by Pubmed.

Quantitative Reproducibility Comparing the distribution of the test accuracies in both experiments, the p-values of the KS-tests are below the significance threshold and thus suggest rejection for all training dimensions except 50%. Therefore, we have statistical evidence for inequality and thus have not reached qualitative reproducibility on Cora. However, for Pubmed, the behavior in both implementations is aligned. The KS-test suggests no significant difference between both implementations. In conclusion, quantitative reproducibility is considered achieved on Pubmed.

Lessons Learned

L8: Conduct Hyperparameter Search As expected, with the hyperparameter values considered earlier KENN does not achieve competitive results on Cora and Pubmed in comparison to the SOTA (PapersWithCode 2023). Hyperparameter tuning significantly affects the results of the experiment. With an appropriate set of hyperparameters for Cora and Pubmed, better results with KENN may be obtained.

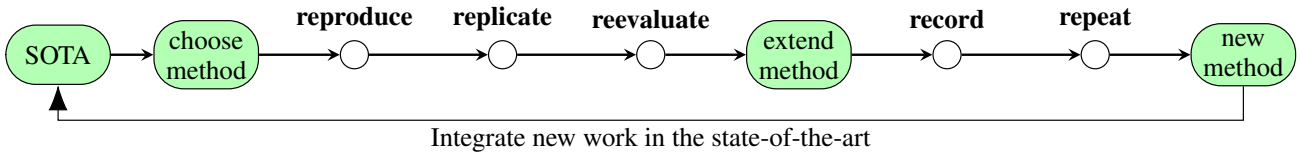


Figure 2: Extending machine learning methods with reproducibility steps.

L9: Verify Results on Other Datasets Overall, the reevaluation experiments show that KENN improves the accuracy on the Cora dataset, but not on the Pubmed dataset. Furthermore, by reevaluation on PubMed, we can detect performance issues and address them, which makes the code applicable to larger datasets in general. As a lesson learned, it can be derived that the application of a model to various datasets can either strengthen result by showing robustness or reveal weaknesses of methods with respect to specific datasets.

Conclusion and Perspective

In this article, we first progressively reproduced, replicated and reevaluated the experiments with KENN before extending it. In terms of qualitative reproducibility, in most experiments the hypothesis that KENN outperforms the base neural network ($H1$) is supported. The relationship between training set size and accuracy ($H2$) is observed for the reproduced experiments but less clear for the replicated and reevaluated experiments. We summarize all the lessons learned in Table 7.

In a broader context, the progressive approach in this work can be incorporated in a general workflow of extending a related method in machine learning while considering reproducibility steps, as illustrated in Fig. 2. While it is tempting to directly jump from a task to the development of a new method, the inclusion of the reproduce, replicate and reevaluate steps can increase the trustworthiness in experimental results. To the three steps conducted in this work, *record* and *repeat* can be added to better integrate the new (extended) method in the SOTA. Recording provides a sufficiently detailed documentation to make the new method accessible and reproducible to the community. Repeating the experiment can serve as a self-check of documentation and/or automation.

Fig. 2 displays an ideal situation in which the path to extension is flawless. In reality, each step may fail and lead the design back to the previous action. However, it helps to isolate the cause of failure at the earliest possible stage.

Further, we note that KENN predominantly satisfies common reproducibility guidelines. However, we encounter difficulties in reproducing, replicating and reevaluating this method. This shows that even though guidelines impose an additional burden for developers, easily and clearly reproducible work reduces the effort required for extending previous work and therefore accelerates advancements. In this sense, reproducibility guidelines should be continuously evaluated and modified (if needed) to ensure that they effectively serve their purpose. From our point of view in the context of this work, reproducibility checklists could improve by integrating some of our lessons learned.

Finally, common reproducibility guidelines are mainly oriented towards reproduction, which ensures that published results correspond to those obtained from the code. Given strong automation, this may only require to clone and run. Even if replication and reevaluation outweigh the effort of reproduction, in return, they provide more insight into the details of an implementation or a method. Moreover, reimplementations make methods more available to the community. In this sense, efforts towards replication and reevaluation should continue to be encouraged, e.g. through reproducibility tracks or challenges (NeurIPS 2019; ICLR 2019).

Reproduction	**L1**	Provide a precise identification of the software environment, source code and dataset
	L2	Automate experiment execution
	L3	Provide contact information and be reachable
	L4	Distinguish different experiments
Replication	**L5**	Document hyperparameters exhaustively
	L6	Provide clear procedures to check claims
	L7	Define standards to evaluate reproducibility
Reevaluation	**L8**	Conduct hyperparameter search
	L9	Verify results on other datasets

Table 7: Summary of the lessons learned.

Experiment Material

The reimplementation of the experiments is available at a dedicated Git repository[4] under the hash `a894cae297b47f6d1acfa8e8dab99603f7b5e996`. It contains the following elements:

- the source code,
- the execution instructions,
- the software requirements,
- the result evaluation script,
- the raw files of the experiment results.

Further, the adaptions of the initial implementation to Pubmed and Cora is included.

Acknowledgements

We are grateful to the authors of KENN for their help. This research has been partially supported by the ANR project GraphRec (ANR-23-CE23-0010) and by the MIAI Knowledge communication and evolution chair (ANR-19-P3IA-0003).

[4]https://gitlab.inria.fr/tyrex-public/reproducibility-aaai24

References

AAAI. 2023. Reproducibility Checklist. https://aaai.org/conference/aaai/aaai-23/reproducibility-checklist/.

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

ACM. 2016. Association for Computing Machinery. Artifact review and badging, 1.1. https://www.acm.org/publications/policies/artifact-review-badging.

Alahmari, S. S.; Goldgof, D. B.; Mouton, P. R.; and Hall, L. O. 2020. Challenges for the Repeatability of Deep Learning Models. *IEEE Access*, 8: 211860–211868.

Ankit, A.; Ambekar, S.; Varadharajan, B.; and Alence, M. 2022. [Re] counterfactual generative networks. *ReScience C*, 8(#2).

Biewald, L. 2020. Experiment Tracking with Weights and Biases. Software available from wandb.com.

Blanco, W.; Lopes, P.; Abner de Souza, A.; and Mascagni, M. 2020. Non-replicability circumstances in a neural network model with Hodgkin-Huxley-type neurons. *Journal of computational neuroscience*, 48(3): 357–363.

Chen, B.; Wen, M.; Shi, Y.; Lin, D.; Rajbahadur, G. K.; and Jiang, Z. M. J. 2022. Towards Training Reproducible Deep Learning Models. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, 2202–2214. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392211.

Chollet, F.; et al. 2015. Keras. https://keras.io.

Daniele, A.; and Serafini, L. 2019. Knowledge Enhanced Neural Networks. In *Proc. Pacific Rim Conference on Artificial Intelligence*, number 11670 in Lecture notes in computer science, 542–554. ISBN 978-3-030-29907-1.

Daniele, A.; and Serafini, L. 2020. Neural Networks Enhancement through Prior Logical Knowledge. *ArXiv*, abs/2009.06087.

Daniele, A.; and Serafini, L. 2023. Knowledge Enhanced Neural Networks for Relational Domains. In Dovier, A.; Montanari, A.; and Orlandini, A., eds., *AIxIA 2022 – Advances in Artificial Intelligence*, 91–109. Cham: Springer International Publishing. ISBN 978-3-031-27181-6.

Diligenti, M.; Gori, M.; and Saccà, C. 2017. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244: 143–165. Combining Constraint Solving with Mining and Learning.

Drummond, C. 2009. Replicability is not reproducibility: nor is it good science. http://cogprints.org/7691/.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Forde, J. Z.; and Paganini, M. 2019. The Scientific Method in the Science of Machine Learning. *ICLR 2019 workshop, May 6, New Orleans*.

Garnelo, M.; and Shanahan, M. 2019. Reconciling deep learning with symbolic artificial intelligence: representing objects and relations. *Current Opinion in Behavioral Sciences*, 29: 17–23. Artificial Intelligence.

Goodman, S.; Fanelli, D.; and Ioannidis, J. 2016. What does research reproducibility mean? *Science translational medicine*, 8(341).

Gundersen, O. E. 2020. The Reproducibility Crisis Is Real. *AI Magazine*, 41(3): 103–106.

Gundersen, O. E.; Gil, Y.; and Aha, D. W. 2018. On Reproducible AI: Towards Reproducible Research, Open Science, and Digital Scholarship in AI Publications. *AI Magazine*, 39(3): 56–68.

Gundersen, O. E.; and Kjensmo, S. 2018. State of the Art: Reproducibility in Artificial Intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep Reinforcement Learning That Matters. In *Proceedings of AAAI'18/IAAI'18/EAAI'18 conferences in Artificial Intelligence*. AAAI Press. ISBN 978-1-57735-800-8.

ICLR. 2019. ICLR Reproducibility Challenge. Second Edition, 2019. https://www.cs.mcgill.ca/~jpineau/ICLR2019-ReproducibilityChallenge.html.

ICML. 2023. Paper Guidelines. https://icml.cc/Conferences/2023/PaperGuidelines.

IJCAI. 2022. Reproducibility guideline. https://ijcai-22.org/reproducibility/.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In Bengio, Y.; and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lones, M. A. 2023. How to avoid machine learning pitfalls: a guide for academic researchers. arXiv:2108.02497.

Lu, Q.; and Getoor, L. 2003. Link-Based Classification. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, 496–503. AAAI Press. ISBN 1577351894.

Lucic, M.; Kurach, K.; Michalski, M.; Gelly, S.; and Bousquet, O. 2018. Are GANs Created Equal? A Large-Scale Study. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Manninen, T.; Aćimović, J.; Havela, R.; Teppola, H.; and Linne, M.-L. 2018. Challenges in Reproducibility, Replicability, and Comparability of Computational Models and Tools for Neuronal and Glial Networks, Cells, and Subcellular Structures. *Frontiers in Neuroinformatics*, 12.

Marra, G.; Diligenti, M.; Giannini, F.; Gori, M.; and Maggini, M. 2020. Relational Neural Machines. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*, volume 325, 1340–1347. IOS Press.

Massey, F. 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253): 68–78.

NeurIPS. 2019. NeurIPS 2019 Reproducibility Challenge. https://reproducibility-challenge.github.io/neurips2019/task/.

PapersWithCode. 2023. Node Classification. https://paperswithcode.com/task/node-classification. Leaderboards.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Pham, H. V.; Qian, S.; Wang, J.; Lutellier, T.; Rosenthal, J.; Tan, L.; Yu, Y.; and Nagappan, N. 2021. Problems and Opportunities in Training Deep Learning Software Systems: An Analysis of Variance. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, 771–783. New York, NY, USA: Association for Computing Machinery. ISBN 9781450367684.

Piantadosi, G.; Marrone, S.; and Sansone, C. 2019. On Reproducibility of Deep Convolutional Neural Networks Approaches.

Pineau, J.; Vincent-Lamarre, P.; Sinha, K.; Lariviere, V.; Beygelzimer, A.; d'Alche-Buc, F.; Fox, E.; and Larochelle, H. 2021. Improving Reproducibility in Machine Learning Research (A Report from the NeurIPS 2019 Reproducibility Program). *Journal of machine learning research*, 22(164): 1–20.

Plesser, H. 2018. Reproducibility vs. replicability: a brief history of a confused terminology. *Frontiers in neuroinformatics*, 11.

PyTorch. 2022. Reproducibility. https://pytorch.org/docs/stable/notes/randomness.html.

Raff, E. 2019. A Step Toward Quantifying Independently Reproducible Machine Learning Research. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Rougier, N. P.; Hinsen, K.; Alexandre, F.; Arildsen, T.; Barba, L. A.; Benureau, F. C.; Brown, C. T.; de Buyl, P.; Caglayan, O.; Davison, A. P.; Delsuc, M.-A.; Detorakis, G.; Diem, A. K.; Drix, D.; Enel, P.; Girard, B.; Guest, O.; Hall, M. G.; Henriques, R. N.; Hinaut, X.; Jaron, K. S.; Khamassi, M.; Klein, A.; Manninen, T.; Marchesi, P.; McGlinn, D.; Metzner, C.; Petchey, O.; Plesser, H. E.; Poisot, T.; Ram, K.; Ram, Y.; Roesch, E.; Rossant, C.; Rostami, V.; Shifman, A.; Stachelek, J.; Stimberg, M.; Stollmeier, F.; Vaggi, F.; Viejo, G.; Vitay, J.; Vostinar, A. E.; Yurchak, R.; and Zito, T. 2017. Sustainable computational science: the ReScience initiative. *PeerJ Computer Science*, 3: e142.

Sinha, K.; Dodge, J.; Luccioni, S.; Forde, J.; Raparthy, S.; Pineau, J.; and Stojnic, R. 2022. ML Reproducibility Challenge 2021. *ReScience C*, 8(#48).

Tian, Y.; Ma, J.; Gong, Q.; Sengupta, S.; Chen, Z.; Pinkerton, J.; and Zitnick, C. L. 2022. ELF OpenGo: An Analysis and Open Reimplementation of AlphaZero. arXiv:1902.04522.

Yang, Z.; Cohen, W. W.; and Salakhutdinov, R. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 40–48. JMLR.org.